A Prototype Curriculum For The Study Of Software Management

Dan Shoemaker, (Email: shoemadp@udmercy.edu), University of Detroit Mercy
Gregory Ulferts, (Email: ulfertgw@udmercy.edu), University of Detroit Mercy
Antonio Drommi, (Email: drommi@udmercy.edu), University of Detroit Mercy

ABSTRACT

The discipline of Software Management, which is a new and potentially meaningful direction for information technology (IT) education, is presented for the first time in this article. Software Management is a curriculum model, which specifically addresses the productivity and quality issues that have arisen in IT. It is distinguished from the traditional disciplines of Computer Science, Software Engineering and Information Science by its body of knowledge, which focuses explicitly on building strategic governance infrastructures rather than technical artifacts. This article presents curricular recommendations for each traditional discipline and uses these to illustrate Software Management's unique role and value. It also presents a conceptual framework and justification, which will assist educators in curriculum development and design issues.

INTRODUCTION

Information technology (IT) is a heterogeneous field, which comprises everything from hardware to strategic management. As such it would probably be as correct to say that quantum physics, which underwrites chip design, is as appropriate to the body of knowledge as macroeconomics. The point being that, each subject is germane to a particular role and purpose. But because of differences in academic preparation and practical issues of student interest and aptitude these topics don't fit together in the same curriculum.

Accordingly, one goal of curriculum design is to distill content in such a way that it embodies a coherent set of practical elements, which will suitably prepare a student to function effectively within some defined area of value. In our case we are addressing the need for focused study of the emerging body of knowledge in the field of strategic management for IT. The concepts that this subject encompasses are derived primarily from lessons learned in advanced technology organizations over the past fifteen years. And they are specifically intended to make the outcomes of IT processes repeatable, as well as eliminate managerial experience as the sole determinant of project success. That is an important issue for business and society as a whole, because the evidence is clear that in the case of *new development* (e.g., something outside of the experience of the manager) or *inexperienced managers*, IT's performance is consistently abysmal[1]. Given this, our article presents the justification, logic and structure of a curricular model, which explicitly centers on providing students with the necessary knowledge in strategic IT management.

This is essentially a new type of study, in that it is targeted on producing a student who specializes in creating governance structures rather than technical products. Its aim is to provide the know-how necessary to allow a practitioner to build a complete strategic management infrastructure for any IT work situation. As such, the knowledge base embodies a different perspective than that of the traditional areas of IT education. Furthermore, because that knowledge is rooted in behavior rather than science it is not derived from natural principles and theory. Instead it is based on lessons learned from industry, which are generally conveyed by professional standards.

[1] Readers interested in detailed proof should refer to Humphrey (1994), Jones (1997), Paulk (1999 and 2000) and Jones (2001) as well as the KPMG and Construx studies cited in the references (among others).

Support for this point of view has evolved over the past fifteen years out of the work of various industry bodies such as the International Standards Organization (ISO) and the Software Engineering Institute (SEI). Those agencies have promulgated large comprehensive models of strategic "best practice", which are intended to be used as templates to define and build a complete and "ideal" IT organization. Generically, these are called "umbrella" standards (or models) to differentiate them from conventional ones. Umbrella standards serve the same purpose within the domain of best practice as theory does for science, in the sense that they comprehensively define and provide the basis to explain the phenomena of a subject field. And since they contain the industry's most expert advice they also provide validated knowledge about a given topic. Therefore, they may be rightly employed as the reference for defining substantive solutions to such hard to capture management concerns as infrastructure development (ISO/IEEE 12207), performance assessment (ISO 15504), or process improvement (ISO 9000, or CMM). Alternatively since these frameworks are meant to be authoritative within the profession, their embedded concepts can provide both the basis and the means of validation for an academic study. Essentially, the goal of such a discipline is to encapsulate and convey the complete set of principles of professional best practice as understood and documented by various legitimate standards bodies and generally embodied in a range of national and internationally recognized umbrella standards.

Because its aim is to produce graduates who will function upon matriculation at the management level this is unavoidably a graduate program. We presume technical competence with little or no direct management experience. However, given that the body of knowledge represents best practice, any manager at any level could theoretically benefit from the thorough grounding that this program of study provides. Having said that, our intake target is a technical worker, perhaps a programmer/analyst, who is ready to move up into the manager role for their team or project. Because of corporate hiring practices these people are usually graduates of a computer science, or software engineering undergraduate program. However, because the body of knowledge is essentially self-contained (within the standards that capture it) we have also worked successfully with students from other backgrounds as long as they have substantive practical experience doing IT work. The end result is a comprehensive mastery of professional best practice in IT management. Graduates are expected to exhibit a long-term personal understanding centered on the rational deployment of integrated technical processes to support all aspects of overall business functioning.

The rest of this article outlines an approach that meets the assumptions itemized above. The program is called Software Management, which is the generally accepted term for this area of study. But, terminology notwithstanding, the purpose and intent is to produce graduate students who have mastered all of the knowledge and skills necessary to enable them to manage and IT organization based on expert principles.

DISCIPLINARY MODEL

Fundamental Concepts and their origin

Although several of these large standard frameworks have appeared over the past decade, they are very similar in terms of their constituent elements[2]. It is that universal commonality, which supplies the basis for defining the curricular elements of the Software Management program with some certainty. Four collective, highly correlated concepts were derived from these models: *abstract representation of process, process oversight, process control,* and *optimization of IT processes.* These are either employed (in the case of abstraction and optimization) or installed (in the case of oversight and control) for the purpose of creating governance frameworks for individual IT operations. In essence, all of the umbrella standard models (and our curriculum) are founded on the assumption that oversight and control are the fundamental conditions of management best practice and that the process itself is developed and tailored through abstraction and must be optimized.

Readers will probably note that these principles are also important elements in other IT related studies. The distinction rests on the fact that we employ them strictly for the purpose of developing, or maintaining an explicit governance framework tailored to fit the functional requirements of a particular IT organization. No other discipline addresses rational, strategic organizational process development in this manner. In the case of the other studies these principles are employed for other legitimate purposes, primarily to create computer, or system artifacts and to

[2] ISO 9000-3 (1993), CMM v1.1 (1993), TickIT (1994), ISO 12207 (1995), ISO 15504 (aka SPICE, 1998)

harmonize computer systems with organizational systems for the purposes of decision support. The substantial difference between these two aims (e.g., governance system versus computer system development) is an important distinction to keep in mind when reading the rest of this.

Abstract Representation of Process

Abstract representation of process serves the purposes of organizational process development by providing the specific means for understanding and describing the problem space. It is also the foundation for tailoring an umbrella standard into a particular representation within the specific context of a given organization, or in simple terms for "drawing the blueprint". Abstraction supports all aspects of tailoring as explicitly specified in models such as IEEE 12207, CMM and 15504. In addition, it provides the basis for the management activities implicit under the concepts of oversight and control. Since abstraction clearly drives the definition and design of computer systems and artifacts (it underlies the first two stages in the waterfall) it is by far the best understood and recognized aspect of the curriculum. Moreover the principles and techniques of this curricular element (including the use of UML) have been borrowed directly from other studies.

Oversight and Control

Oversight and control are well understood in the world of manufacturing and general business as the twin elements of applied management. It is commonly recognized that the presence or absence of these two elements determines project success. Practically speaking, in order to manage an operation, its activities have to be both visible and controllable. Since most IT work is either too complex or too creative and abstract to be directly observable this presents a problem. As a consequence, the technical staff in most cases usually knows more about the actual status of a project than the manager, who is the one accountable for its success. The same is true of control. IT organizations are exceedingly flat because, as it is currently conducted, IT work is creative and heuristic, not logical and linear. Accordingly, most of the Key Process Areas at Level Two CMM (e.g., the entry level) concentrate on establishing oversight and control of the IT operation. This is also true with IEEE 12207 in that the main lifecycle supporting processes are documentation, configuration management and SQA, which all directly serve these purposes.

Optimization

Finally, definition and the organizational mechanisms to establish and enforce visibility and control are insufficient if they do not correct defects and make the operation continuously improving. As such, every practical organizational model has to be optimized. Any person with a passing acquaintance with the "quality" universe does not have to have the importance of optimization explained to them. It is the only underlying purpose of CMM (the end product of which is called the "optimized" level) as it is with ISO 15504 (which is the assessment standard for IEEE12207).

THEMATIC AREAS

These concepts are embodied in six thematic areas, which taken as an integrated set, also comprise the attributes necessary to create rational processes. Three of these are plainly traceable to their functional origin (Modeling, Control Systems, Monitoring Systems). Process Engineering is the generic title for process development and optimization. Since Assessment with Measurement provides the basis for organizational monitoring and control, as well as for optimization (at any level of capability), it could be considered a primitive rather than a theme. But because it is a major component of three other concepts it was decided to treat it across the board in the curriculum at the next level of definition (e.g., application). Logically, part of the program is built around the pragmatics of product development, so the Construction and Reuse theme is a concession to reality. However, even in such classes, the pedagogy approaches the traditional content from the direction of the models that dictate proper practice. This includes a heavy emphasis on re-engineering and reuse.

Figure 1 - Thematic Areas

1.	Applied Abstract Modeling
2.	Process Engineering
3.	Organizational Control Systems
4.	Organizational Monitoring Systems
5.	Assessment with Measurement
6.	Construction and reuse

The curricular strategy is hierarchical. The focus is top-down from models down to methods for developing the specific implementation. The presentation centers on ways to apply professional standards and umbrella frameworks. This is embodied in the five course subject areas of:

Figure 2 - Course Subject Areas

1.	Project Management
2.	Object Oriented Modeling
3.	Specification and Design (principles and methods)
4.	Software Quality Assurance and Metrics
5.	Strategic Process Development and Configuration Management

These subject areas plus germane simulated real-world experience introduce the relevant principles of the discipline. It allows students to develop and internalize their own comprehensive understanding as well as formulate a personal model of the body of knowledge. There are technical courses offered simply to satisfy a range of potential interests. Students opt for these based on their aptitude and interest, varying from individual to individual. However, the body of knowledge in Software Management is delivered in the core, which is required for all students.

Table One lists the sixteen course elements, which comprise the current Software Management curriculum. It is believed (and can be easily proven based on cross references to the various conceptual models) that this array represents a correct set of integrated knowledge essentials, which will underwrite the creation of effective and efficient, IT governance infrastructures. The number in parentheses roughly references the thematic area (see legend, and content areas), which each course embodies.

As explained earlier, there is basic knowledge about program construction and technical work built into the value added (e.g., elective) areas, specifically Databases (elective 7), Networking (elective 8), GUI Development (elective 5), and Distributed System Development (elective 9). However, the apparent overlap between Object Oriented Programming (core 2) and any course in Object Orientation contained in software engineering or computer science curriculum is deceptive. The traditional version of this course is designed to teach object oriented programming techniques. The Software Management course is designed to teach object modeling and UML principles for the purpose of process understanding and architecture.

Table 1 - Body of Knowledge Software Management

Core			
Project Management	2, 3, 4, 5	Software project management subject area	
Object Oriented Programming	1, 2, 6	Modeling and methods subject area	
Software Requirements Specification	1, 2, 6	Specification and design subject area	
Software Quality Assurance and Testing	2, 3, 4, 5	SQA and metrics subject area	
Strategic Software Process Management	2, 3, 4, 5	Quality and CM subject area	
Electives (6 required)			
Software System Documentation	2	Data Base Design	1, 6
Software Design and Construction	1, 6	Network and Network Management	6
Metrics and Models for Software Management	2, 3, 5, 5	Distributed System Development	1, 6
Software Lifecycle Documentation	1, 2, 3, 4, 5	Leadership in Assessment	3, 4, 5
Graphical User Interface Development	6	Internal Audit	3, 4, 5
Software Maintenance Using Cobol	6	Legend: 1. Modeling, 2. Process Engineering, 3. Control, 4. Monitoring, 5. Measurement, 6. Construction	

APPLICATION OF BODY OF KNOWLEDGE TO INDUSTRY

From the standpoint of application to the real world, the conceptual origins of this entire group of courses either lies within a CMM Key Process Area (KPA) or an IEEE 12207.0 process. For instance Software Requirements Specification (core course 3) is employed both in the development primary process as the first stage in the waterfall and in the acquisition-supply processes as the basis for the contract and subsequent monitoring and control activities. It is also the fundamental building block of configuration management baselines, which makes it an essential element of the maintenance process (IEEE 12207.0 process 5.5).

Strategic Software Management (core course 5) provides the framework for the definition and tailoring of the infrastructure (IEEE 12207.0 process 7.2) as well as the employment of all of the processes in the lifecycle and the conventional models for continuous improvement (IEEE 12207.0 process 7.3). The Metrics and Models course (elective 4) establishes the CMM Organization Process Definition KPA as well as the Measurement and Analysis Common Feature. Both of these are keys to achieving a managed organization. In conjunction with this, Lifecycle Documentation (elective 6) creates a schema that allows for quantitative management of the IT organization (as specified in the lifecycle data element definitions of IEEE 12207.1). The two Assessment courses (electives 10 and 11) reflect the emphasis on organizational assessment requisite for building an effective monitoring and feedback system for process assurance. And finally, the Maintenance course (elective 6) details the activities of the maintenance lifecycle process (IEEE 12207.0 process 5.5) as well as examines the reengineering of legacy software.

DEFINING SOFTWARE MANAGEMENT'S SPECIFIC ROLE

It seems useful to compare the Software Management curriculum with the curricula of the traditional computer disciplines, both in order to explore how it might be incorporated into an existing program and for the purpose of highlighting its unique (and we believe important) role. As we said earlier, the primary difference between this study and the established disciplines lies in its focus, which is on the creation of rational governance infrastructures rather than classic computer applications. This can be easily demonstrated by looking at the model curricula of the three commonly accepted studies.

As a mechanism to aid this comparison, we would like to employ the standard model for computer organization and architecture. As many readers may know, computers are understood on four hierarchical levels from circuits (the lowest level) through logic (the second) and programming (the third) through systems (the highest). Properly, *Computer Science*, which is by far the oldest of the computer disciplines, concentrates at levels one, two and three. This study originated in the 1950s in departments of mathematics and philosophy and it has never strayed very far from its roots. Thus it is oriented toward fostering the study of linear logic and the various implementations derived from that. In that respect its goals are the least similar to software management and its body of knowledge is

the least aligned with it. The last model curriculum was defined in 1991 however it is currently under revision as CC2001. Table Two lists the 14 topic areas currently being considered.

Table 2 - The Computer Science body of knowledge CC2001

1.	Discrete Structures
2.	Programming Fundamentals
3.	Algorithms and Complexity
4.	Programming Languages
5.	Architecture and Organization
6.	Operating Systems
7.	Net-Centric Computing
8.	Human Computing Interaction
9.	Graphics and Visual Computing
10.	Intelligent Systems
11.	Information Management
12.	Software Engineering
13.	Social and Professional Issues
14.	Computational Science

As can be seen there are two topics that are not strictly scientific, "Social and Professional Issues" and "Information Management". These represent issues that have arisen since the inception of the discipline and which the ACM considers to be germane. However, neither of these has a governance focus.

The second discipline is *Information Systems* (sometimes known as MIS). Using our common classification structure, that study focuses primarily in the top two layers (e.g., programming and systems). It overlaps with Computer Science in its concentration at the third level (programming), but unlike Computer Science there is no specific mandate to present content at either the circuit or logic level in any depth (however an overview of these might legitimately appear in the introductory course) and it is not founded on mathematics (as a central discipline). In effect (as the name implies) it centers on external business applications for the computer, rather than the science of it. The general focus of this discipline is on the methods and models for supporting management decision making through the provision of information (the analysis of which might be a highly mathematical process). The most recent version of this curriculum (Table Three) was promulgated in 1997 (IS97) however the sponsors (ACM, AIS, AITP) are working on a revision entitled IS2002, which will incorporate additional perspectives.

Table 3 - IS97 Curriculum Areas and Component Courses

Information System Fundamentals
IS'97.1 Fundamentals of Information Systems
IS'97.2 Personal Productivity with IS Technology
Information Systems Theory and Practice
IS'97.3 Information Systems Theory and Practice
Information Technology
IS'97.4 Information Technology Hardware and Software
IS'97.5 Programming, Data, File and Object Structures
IS'97.6 Networks and Telecommunication
Information Systems Development
IS'97.7 Analysis and Logical Design
IS'97.8 Physical Design and Implementation with DBMS
IS'97.9 Physical Design and Implementation with a Programming Environment
Information Systems Deployment and Management Processes
IS'9710 Project Management and Practice

As can be seen, since they share a mutual focus at the programming level there are a number of topics which might appropriately fit in either a Computer science, or an IS curriculum including IS97.3, IS97.4, IS97.5, IS97.6, IS97.8 and IS97.9. There are also two courses that are appropriate to the system/decision support level (IS97.3 and IS97.7) which are essentially unique to MIS. Finally there is a Project Management course, which would fit in either an MIS, or a Software Management curriculum (IS97.10). Given that the latter topic lies under a general category heading of "Information Systems Deployment and Management Processes" (which characterizes a focus of the software management curriculum) it is possible that other governance-oriented content might be added to later versions of this curriculum. This would cause MIS to overlap more with the focus and intent of Software Management.

Finally, there is S*oftware Engineering*, which currently does not have a defined body of knowledge. A joint working group formulated by IEEE and the ACM is involved in a multi-year project (called the SWEBOK) to formulate such a body of knowledge. These are the ten knowledge areas that have been identified in the Stoneman version (v.07).

Table 4 - The SWEBOK Knowledge Areas (KA)

KA 1: Software Requirements	KA 6: Software Management
Requirements Engineering	Measurement
Requirements Elicitation	Coordination and Management
Requirement Analysis	Initiation and Scope Definition
Requirements Specification	Planning
Requirements Validation	Enactment
Requirement Management	Review and Evaluation
KA 2: Software Design	Project Close-out
Software Design Concepts	Post Closure Activities
Software Architecture	KA 7: Configuration Management
Software Design Quality Evaluation	Management of SCM
Software Design Notation	SCM Identification
Software Design Strategies/ Methods	SCM Control
KA 3: Software Construction	SCM Status Accounting
Linguistic Construction Methods	SCM Auditing
Formal Construction Methods	Release Management and Delivery
Visual Construction Methods	KA 8: Software Processes
KA 4: Software Testing	Basic Concepts and Definition
Basic Concepts	Process Infrastructure
Test Levels	Process Measurement
Test Techniques	Process Definition
Test Related Measures	Qualitative Process Analysis
Management of the Test Process	Process Implementation and Change
KA 5: Software Maintenance	KA 9: Tests & Methods
Introduction to Maintenance	Software Tools
Maintenance Activities	Software Methods
Maintenance Process	KA: 10: Software Quality
Organizational Aspects	Quality Concepts
Problems of Maintenance	Defining SQA and V&V
Maintenance Cost Estimation	Planning for SQA and V&V
Maintenance Measurement	Techniques for SQA and V&V
Maintenance Techniques	Measurement for SQA and V&V

Since software engineering (SWE) is explicitly oriented toward best practice program construction principles and techniques it overlaps with them at the programming level. Thus it embodies a lot of the focus of both MIS and Computer Science in KA1, KA2, and KA3. However, because it considers itself to be an engineering discipline (and it

tends to be located in engineering schools), SWE is very scientific in the body of knowledge that it emphasizes for these three KAs (e.g., the techniques it emphasizes are math based). It is also highly mathematical both in its general conceptual orientation and in the expectations that it has for its entering students. As such, it appears closer in its purposes to Computer Science than MIS. In fact, particularly in the design and programming courses software engineering instruction would be indistinguishable from Computer Science.

Software engineering was created as a discipline from industry concepts designed to support "programming in the large" (an original description of the intent of SWE fostered by SEI). As such, it focuses at the top two layers of computer architecture (programming and systems), which means it shares a similar interest in program design and construction with MIS. Within that domain, the knowledge areas of SWE are arrayed more in the fashion of the Waterfall model of software development. But it is obvious that the purpose and intent of IS97.7 is also served by KA1 and KA2. Where it diverges from the body of knowledge of both Computer Science and MIS on the program construction side (on the surface at least, since much of this content could be embodied in programming courses at the sub-topic level) is in KA4 and KA9. These two KAs represent best practice in the assurance elements of program construction without regard to technique.

In general there is almost no documented overlap between the best practice management content captured by KA6, KA7, KA8 and KA10 and the bodies of knowledge of either Computer Science or MIS. However since it is governance focused, these four areas DO strongly overlap with the body of knowledge of software management. In fact, given the fact that four of ten of SWE's KAs lie expressly within the domain of governance and many typical management functions are interspersed throughout that body of knowledge there is good reason to presume that software management might simply be a subset of software engineering.

In particular, the emphasis on process infrastructure and software management concepts such as configuration management and SQA underwrites the view that good engineering and good processes go hand in hand. Nevertheless, the fact remains that the SWEBOK is dedicated to enhancing software development capability, not advancing the overall competence of the IT organization. Moreover, although software development and deployment is the pivotal element for the overall IT function, it does not specifically require or imply the governance motivated, strategic infrastructure development purpose, which is the sole aim of the discipline of Software Management. Or, in simple terms, the program construction process can be conducted (and frequently is) independent from and without practices that manage it better. Consequently, there is justification for treating Software Management as a discipline separate from Software Engineering.

CONCLUSIONS

Notwithstanding the question of whether Software Management is a separate discipline, the point of this discussion is that there is a clear need for intensively focused education in IT management, wherever it might be located on campus. That is because the concepts associated with IT governance are not the same as those that pertain to the technical aspects of the profession. This is not currently addressed specifically in Computer Science, MIS or Software Engineering. Nor does general business education (as represented by the MBA) offer any of the management principles and perspectives needed to address the unique management concerns of a technical organization.

That would not be important issue if it were not for the fact that although IT has a forty-year history of leadership and innovation in the production of technical goods and services, the evidence is clear that it has to be better managed. For instance, a Standish Group survey of 8,000 projects found that fewer than half of them finish within their allotted schedules and budgets (Construx, 1998). A nationwide survey conducted by KPMG found that the average project was 120% past its schedule and 90% over budget (KPMG, 2001). The outcome of this poor performance is that Strassman (2001) found based on an examination of 1,585 US companies that there was no correlation whatsoever between investment in information technology and profitability. Which leads him to conclude that "*Corporations are finally fed up with IT and they cannot afford escalating costs any more. They always compensated for incompetence by spending more money, but we are out of money now*."

A recent study by PA Consulting (2001) also found that businesses were disillusioned over the return on their IT spending. Which led Tom Jones, a partner with that firm, to conclude that, *"Many companies are frustrated by the practicalities and pressures on their IT resources and find that the impetus for delivering business benefits is frequently lost after the financial investment and immediate project deadlines are met, If IT is to be firmly embedded in companies' strategic thinking, CEOs must take a prominent role* (Fisher, 2001)." It is the emerging recognition that IT work is not a technical operation alone, rather it is a strategic function in which business goals are realized by IT processes, that provides the primary justification for the focused presentation of the body of knowledge in how to do that best. Brent Habig, the head of US-based Tigris Consulting, summarizes this "*It is vital that IT is aligned much more closely to real business goals than in the past. IT is challenged to have a greater understanding of the business to succeed in the new environment. Fundamentally, it is a rewriting of the relationship between IT and business* (Fisher, 2001)."

Given all of this, we believe that a study aimed solely at presenting the current best practice know-how for integrating IT processes into business organizations is a profitable new direction for higher education. Furthermore because of the complexity and the importance of the problem a distinct curriculum deliberately centered on that particular mission is necessary. Moreover, we believe that this body of knowledge is a potentially important and meaningful new direction for any university, particularly in business schools. Nevertheless, the curriculum presented here is distilled from common elements found in the profession as well as all of the other three disciplines. We offer our model as a potential guide that will help educators judge what to include in such a study, or where it fits in their current curricula.

REFERENCES

1. Abran, A., et al. (2000), *Guide to the Software Engineering Body of Knowledge*, *Stone Man version 0.7*, IEEE – ACM.
2. Association for Computing Machinery (December 15, 2001), *Computing Curricula 2001*, ACM-IEEE-CS.
3. Construx Software Builders, website @ www.construx.com, current (9/2001).
4. Davis, G., et al. (1997), *Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems,* Association for Computing Machinery (ACM), Association for Information Systems (AIS), Association of Information Technology Professionals (AITP).
5. Fisher, Andrew, *System Suppliers Face up to New Mood of Realism*, FTIT, December 3, 2001
6. Humphrey, W. (January 2001), "The Changing World of Technology", *Crosstalk*, Volume 14, #1.
7. IEEE (March 1998), *IEEE/IEA 12207.0*, IEEE.
8. IEEE (March 1998), IEEE*/IEA 12207.1*, IEEE.
9. IEEE (March 1998), *IEEE/IEA 12207.2*, IEEE.
10. *Increasing business value with Information Technology*, PA Consulting, 2001.
11. Jones Capers, *Software Project Management in 2001*, "A Survey of the State of the Art", Software Productivity Research, Artemis, Burlington, Mass., 2001.
12. Jones, Capers, *Assessment and Control of Software Risks*, Prentice-Hall: Englewood Cliffs, 1997, NJ.
13. KPMG Technology and Services Group, web site at www.kpmg.ca, current (11/ 2001).
14. McGibbon, T. (1999), *A Business Case for Software Process Improvement Revised*, DoD Data Analysis Center for Software (DACS).
15. Paulk, Mark C, *Toward Quantitative Process Management With Exploratory Data Analysis*, International Conference on Software Quality, Cambridge, MA, 1999.
16. Paulk, Mark C., Dennis Goldenson, David M. White, *Survey of High Maturity Organizations*, SPECIAL REPORT CMU/SEI-2000-SR-002, 2000.
17. Paulk, M., et al. (1993) *Capability Maturity Model, Version 1.1*, Technical Report, Software Engineering Institute, Carnegie-Mellon University.
18. Software Engineering Institute, www.sei.cmu.edu. (current 11/2001).

NOTES